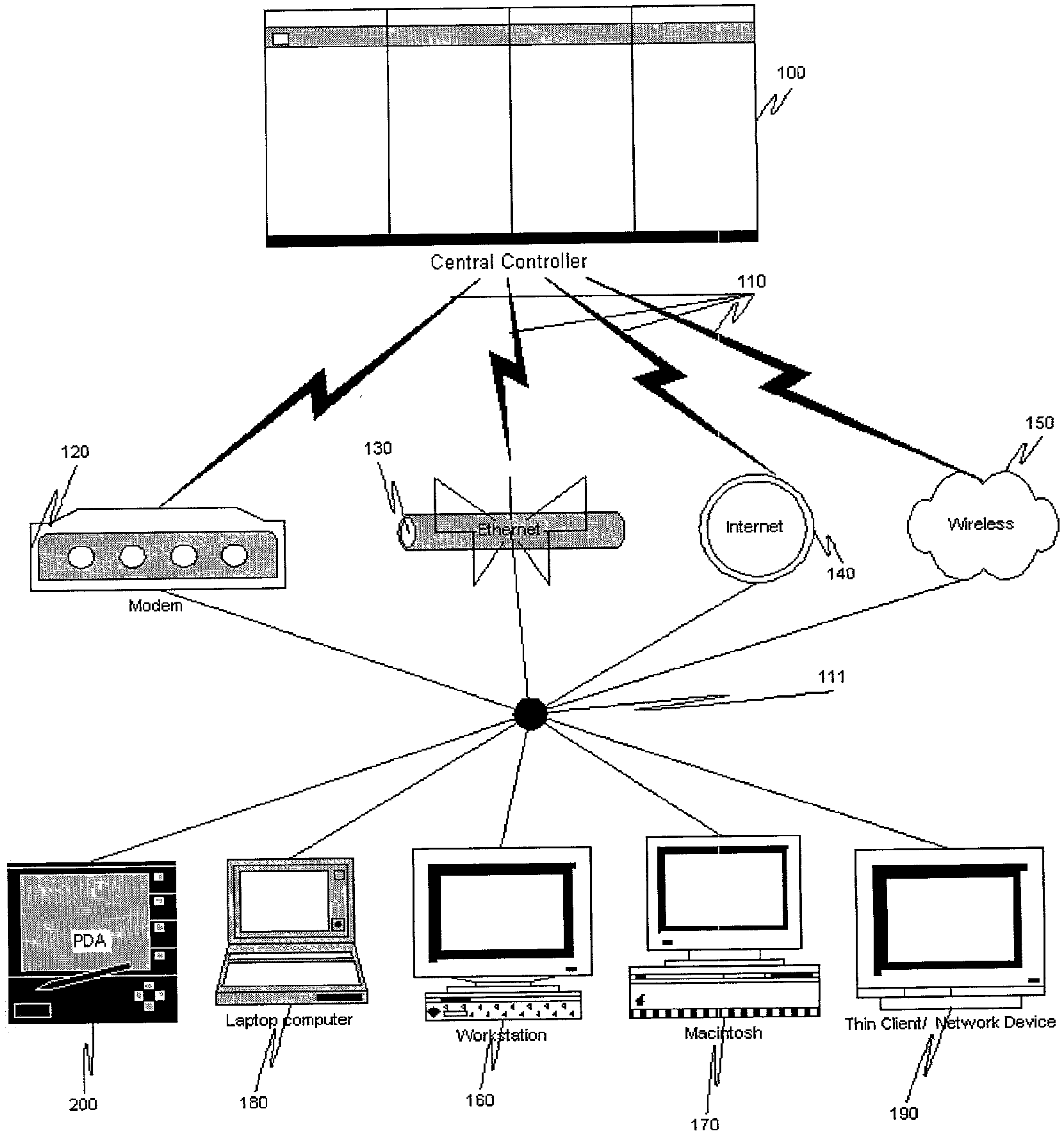

100
Central Controller
110
120
130
150
Modem
Ethernet
Internet
Wireless
140
111
PDA
Laptop computer
Workstation
Macintosh
Thin Client/ Network Device
200
180
160
170
190

FIG. 1

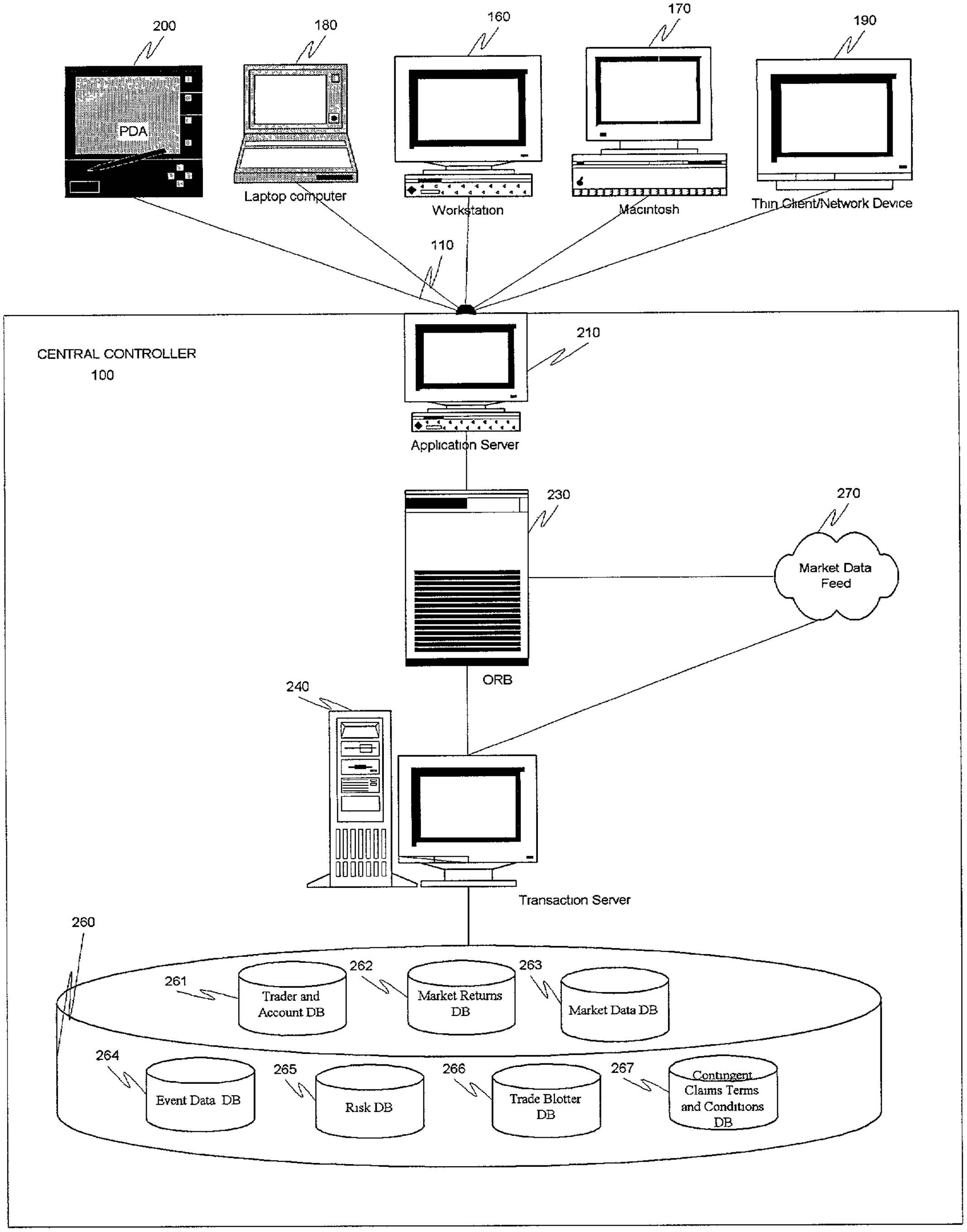
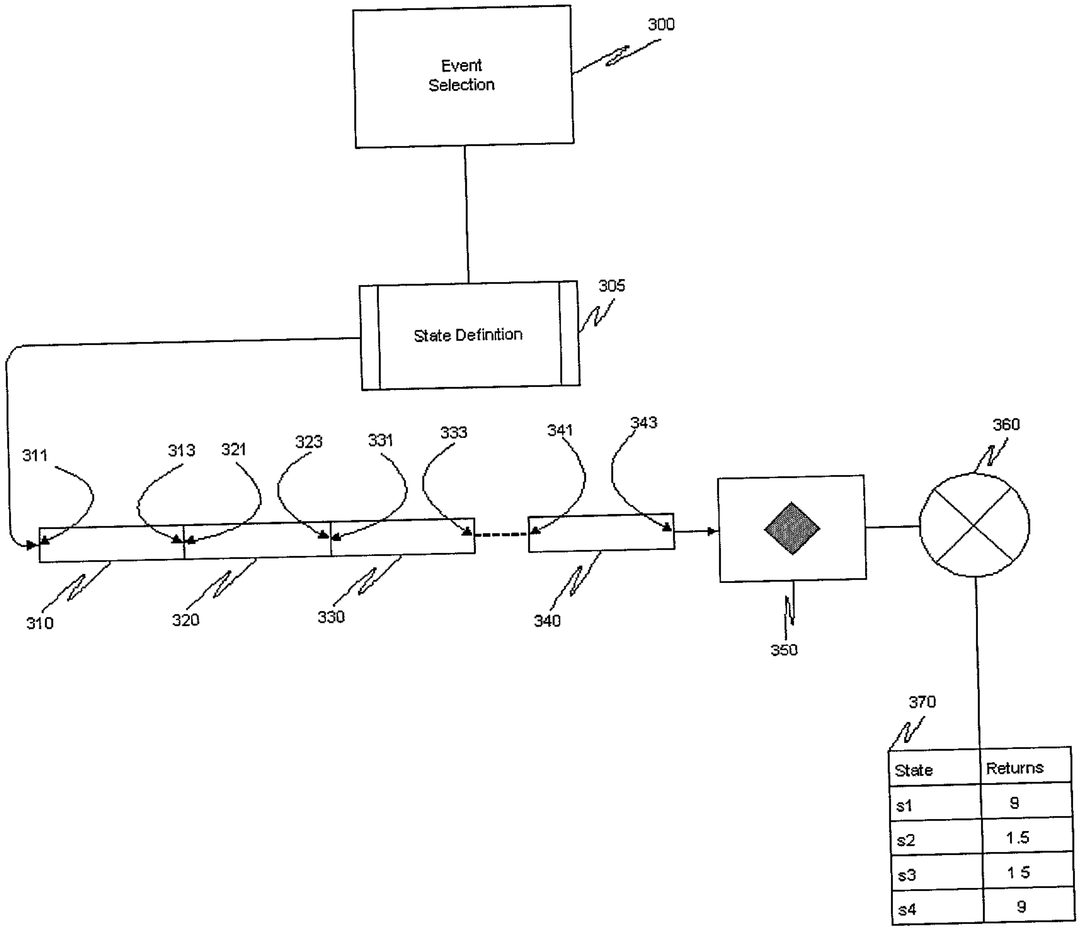

200
PDA
180
Laptop computer
160
Workstation
170
Macintosh
190
Thin Client/Network Device
110
CENTRAL CONTROLLER
100
210
Application Server
230
ORB
270
Market Data Feed
240
Transaction Server
260
261
Trader and Account DB
262
Market Returns DB
263
Market Data DB
264
Event Data DB
265
Risk DB
266
Trade Blotter DB
267
Contingent Claims Terms and Conditions DB

FIG. 2



State	Returns
s1	9
s2	1.5
s3	1 5
s4	9

FIG. 3

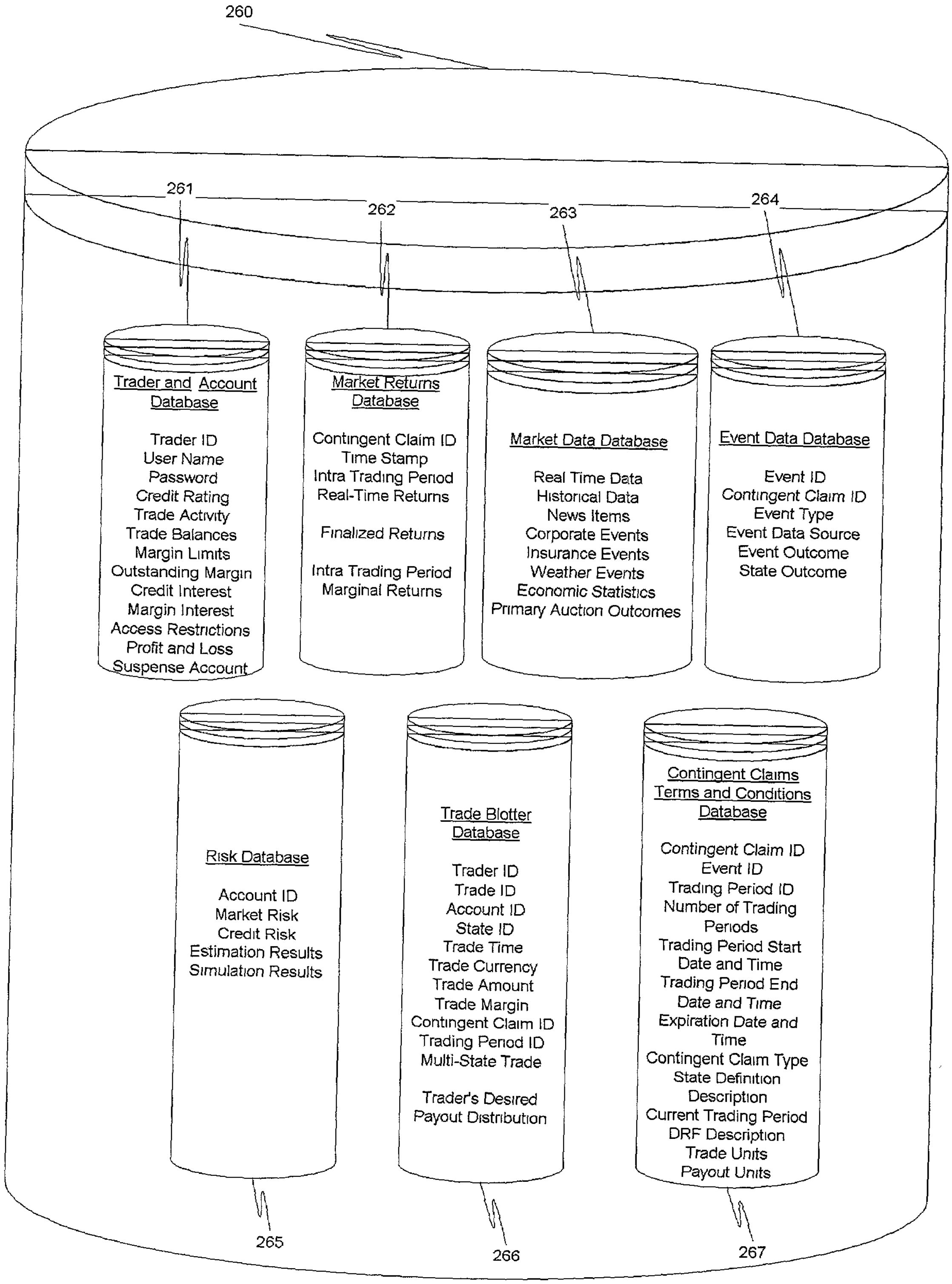

260
261
262
263
264
Trader and Account Database
Trader ID
User Name
Password
Credit Rating
Trade Activity
Trade Balances
Margin Limits
Outstanding Margin
Credit Interest
Margin Interest
Access Restrictions
Profit and Loss
Suspense Account
Market Returns Database
Contingent Claim ID
Time Stamp
Intra Trading Period
Real-Time Returns
Finalized Returns
Intra Trading Period
Marginal Returns
Market Data Database
Real Time Data
Historical Data
News Items
Corporate Events
Insurance Events
Weather Events
Economic Statistics
Primary Auction Outcomes
Event Data Database
Event ID
Contingent Claim ID
Event Type
Event Data Source
Event Outcome
State Outcome
Risk Database
Account ID
Market Risk
Credit Risk
Estimation Results
Simulation Results
Trade Blotter Database
Trader ID
Trade ID
Account ID
State ID
Trade Time
Trade Currency
Trade Amount
Trade Margin
Contingent Claim ID
Trading Period ID
Multi-State Trade
Trader's Desired Payout Distribution
Contingent Claims Terms and Conditions Database
Contingent Claim ID
Event ID
Trading Period ID
Number of Trading Periods
Trading Period Start Date and Time
Trading Period End Date and Time
Expiration Date and Time
Contingent Claim Type
State Definition Description
Current Trading Period
DRF Description
Trade Units
Payout Units
265
266
267

FIG. 4

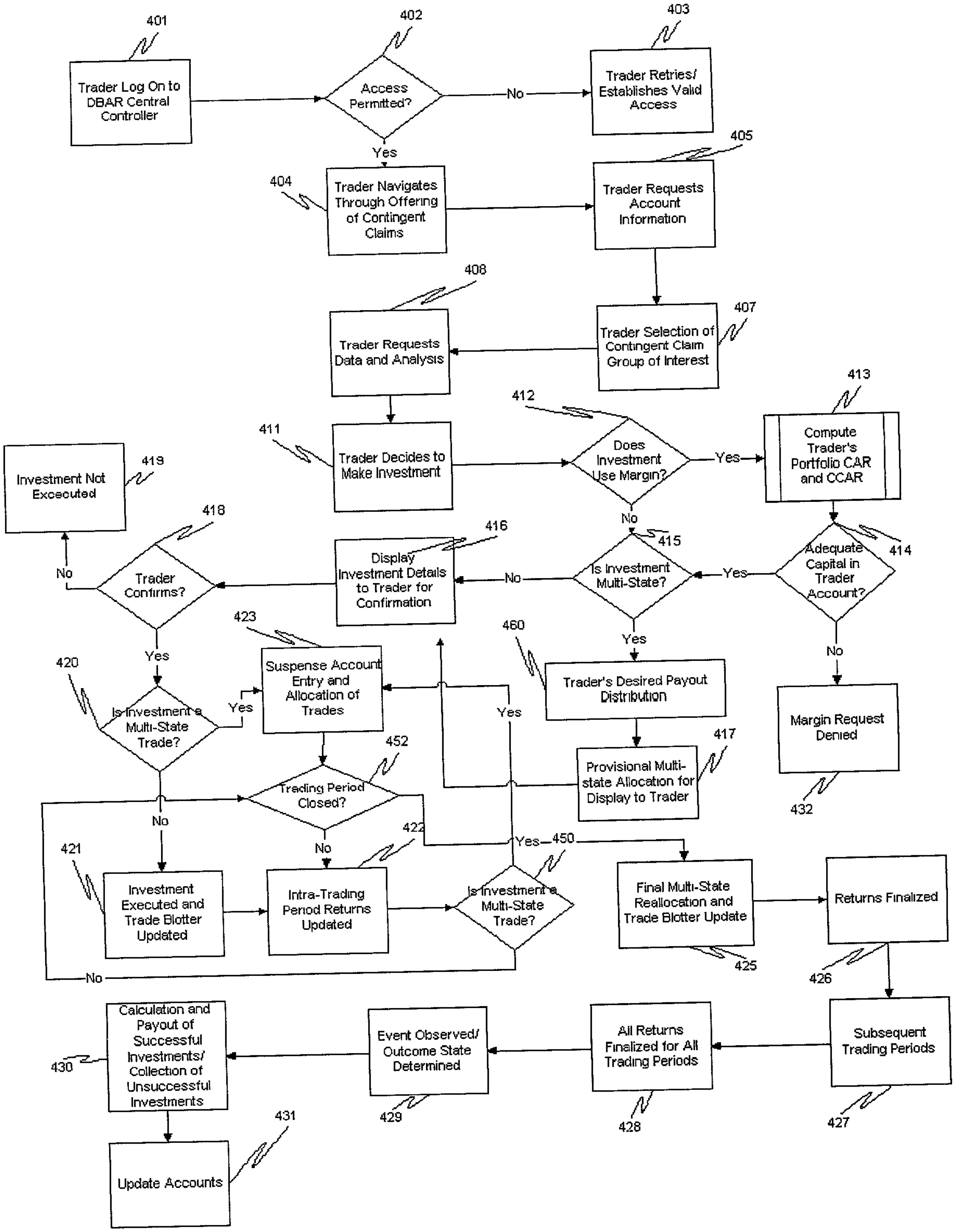
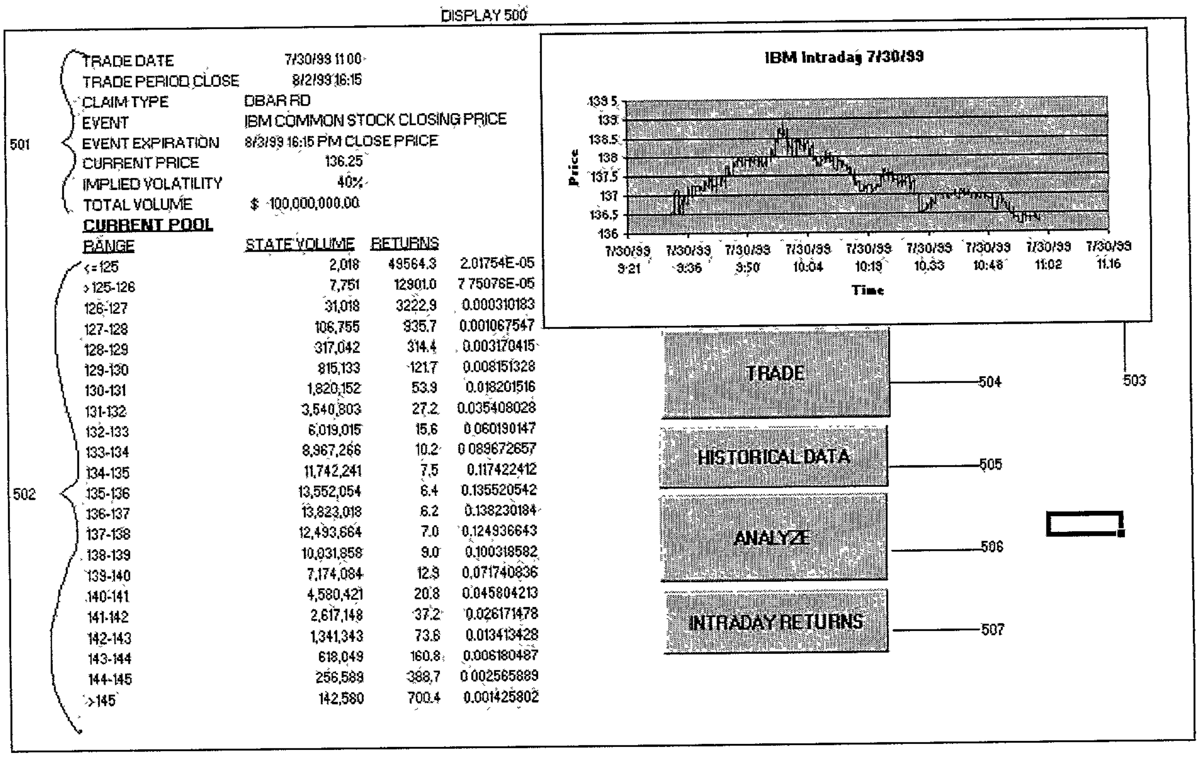

401
Trader Log On to DBAR Central Controller
402
Access Permitted?
No
403
Trader Retries/ Establishes Valid Access
Yes
404
Trader Navigates Through Offering of Contingent Claims
405
Trader Requests Account Information
408
Trader Requests Data and Analysis
407
Trader Selection of Contingent Claim Group of Interest
411
Trader Decides to Make Investment
412
Does Investment Use Margin?
Yes
413
Compute Trader's Portfolio CAR and CCAR
No
419
Investment Not Exceecuted
418
No
Trader Confirms?
416
Display Investment Details to Trader for Confirmation
No
415
Is Investment Multi-State?
Yes
414
Adequate Capital in Trader Account?
Yes
423
Suspense Account Entry and Allocation of Trades
460
Trader's Desired Payout Distribution
No
Margin Request Denied
432
420
Is Investment a Multi-State Trade?
Yes
Yes
452
Trading Period Closed?
417
Provisional Multi-state Allocation for Display to Trader
No
No
421
Investment Executed and Trade Blotter Updated
422
Intra-Trading Period Returns Updated
Yes
450
Is Investment a Multi-State Trade?
Final Multi-State Reallocation and Trade Blotter Update
425
Returns Finalized
426
No
Calculation and Payout of Successful Investments/ Collection of Unsuccessful Investments
430
Event Observed/ Outcome State Determined
429
All Returns Finalized for All Trading Periods
428
Subsequent Trading Periods
427
431
Update Accounts

FIG. 5

DISPLAY 500

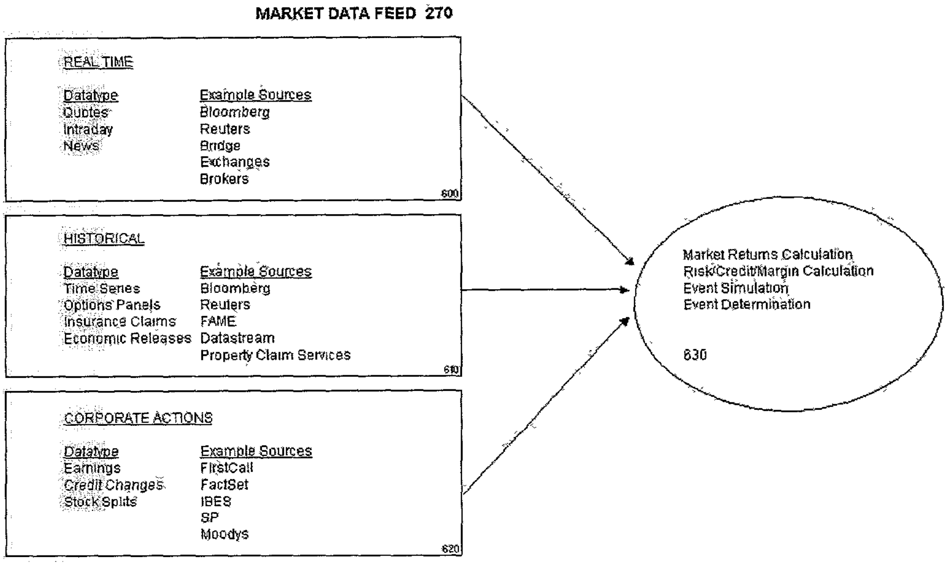


TRADE DATE	7/30/99 11:00
TRADE PERIOD CLOSE	8/2/99 16:15
CLAIM TYPE	DBAR RD
EVENT	IBM COMMON STOCK CLOSING PRICE
EVENT EXPIRATION	8/3/99 16:15 PM CLOSE PRICE
CURRENT PRICE	136.25
IMPLIED VOLATILITY	40%
TOTAL VOLUME	$ 100,000,000.00

CURRENT POOL

RANGE	STATE VOLUME	RETURNS	
<=125	2,018	49564.3	2.01754E-05
>125-126	7,751	12901.0	7.75076E-05
126-127	31,018	3222.9	0.000310183
127-128	106,755	935.7	0.001067547
128-129	317,042	314.4	0.003170415
129-130	815,133	121.7	0.008151328
130-131	1,820,152	53.9	0.018201516
131-132	3,540,803	27.2	0.035408028
132-133	6,019,015	15.6	0.060190147
133-134	8,967,266	10.2	0.089672657
134-135	11,742,241	7.5	0.117422412
135-136	13,552,054	6.4	0.135520542
136-137	13,823,018	6.2	0.138230184
137-138	12,493,664	7.0	0.124936643
138-139	10,031,858	9.0	0.100318582
139-140	7,174,084	12.9	0.071740836
140-141	4,580,421	20.8	0.045804213
141-142	2,617,148	37.2	0.026171478
142-143	1,341,343	73.6	0.013413428
143-144	618,049	160.8	0.006180487
144-145	256,589	388.7	0.002565889
>145	142,580	700.4	0.001425802

FIG. 6



REAL TIME

Datatype	Example Sources
Quotes	Bloomberg
Intraday	Reuters
News	Bridge
	Exchanges
	Brokers

HISTORICAL

Datatype	Example Sources
Time Series	Bloomberg
Options Panels	Reuters
Insurance Claims	FAME
Economic Releases	Datastream
	Property Claim Services

CORPORATE ACTIONS

Datatype	Example Sources
Earnings	FirstCall
Credit Changes	FactSet
Stock Splits	IBES
	SP
	Moodys

FIG. 7

Implied Liquidity Effects: Percentage Changes to Implied State Probabilities Between "Offer"and "Bid" as a Function of Proposed Investment Amount (as a percentage, p, of existing investment)

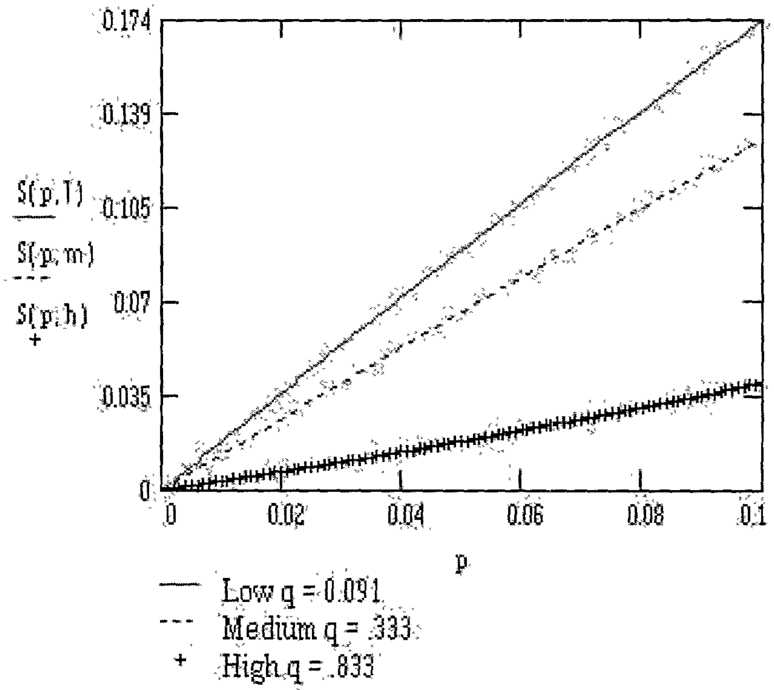


FIG. 8

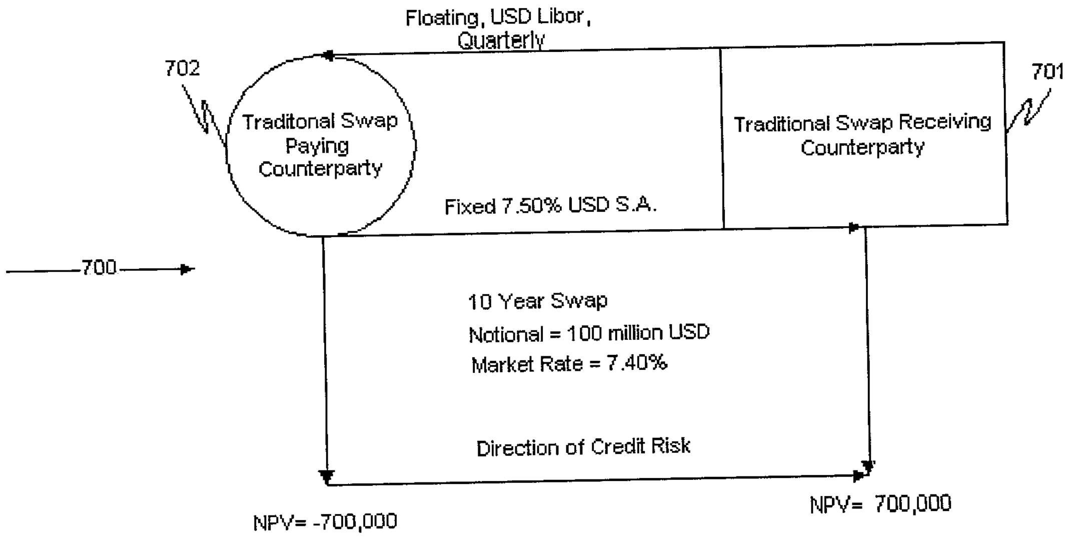

Floating, USD Libor, Quarterly
702
Traditonal Swap Paying Counterparty
Fixed 7.50% USD S.A.
Traditional Swap Receiving Counterparty
701
700
10 Year Swap
Notional = 100 million USD
Market Rate = 7.40%
Direction of Credit Risk
NPV= -700,000
NPV= 700,000

FIG. 9a: Traditional Swap Counterparties

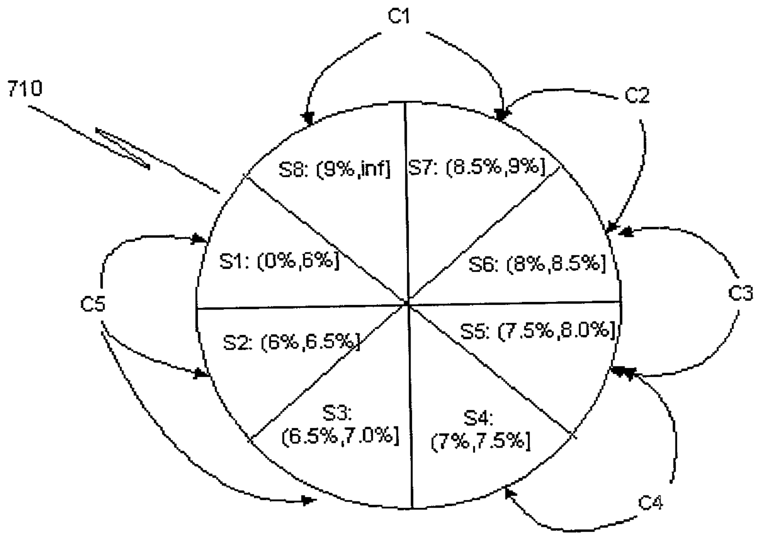


FIG. 9b: Illustrative Trader Relationships In DBAR Contingent Claims

720

States for Swap Rate

Counterparty and Credit Rating

	S1	S2	S3	S4	S5	S6	S7	S8
C1,AAA							50,000	100,000
C2, AA						40,000	25,000	
C3, AA					100,000	60,000		
C4, A+				150,000	100,000			
C5, A	100,000	50,000	80,000					

FIG 9c: Margin Loans by Trader, Credit Rating, and Defined State

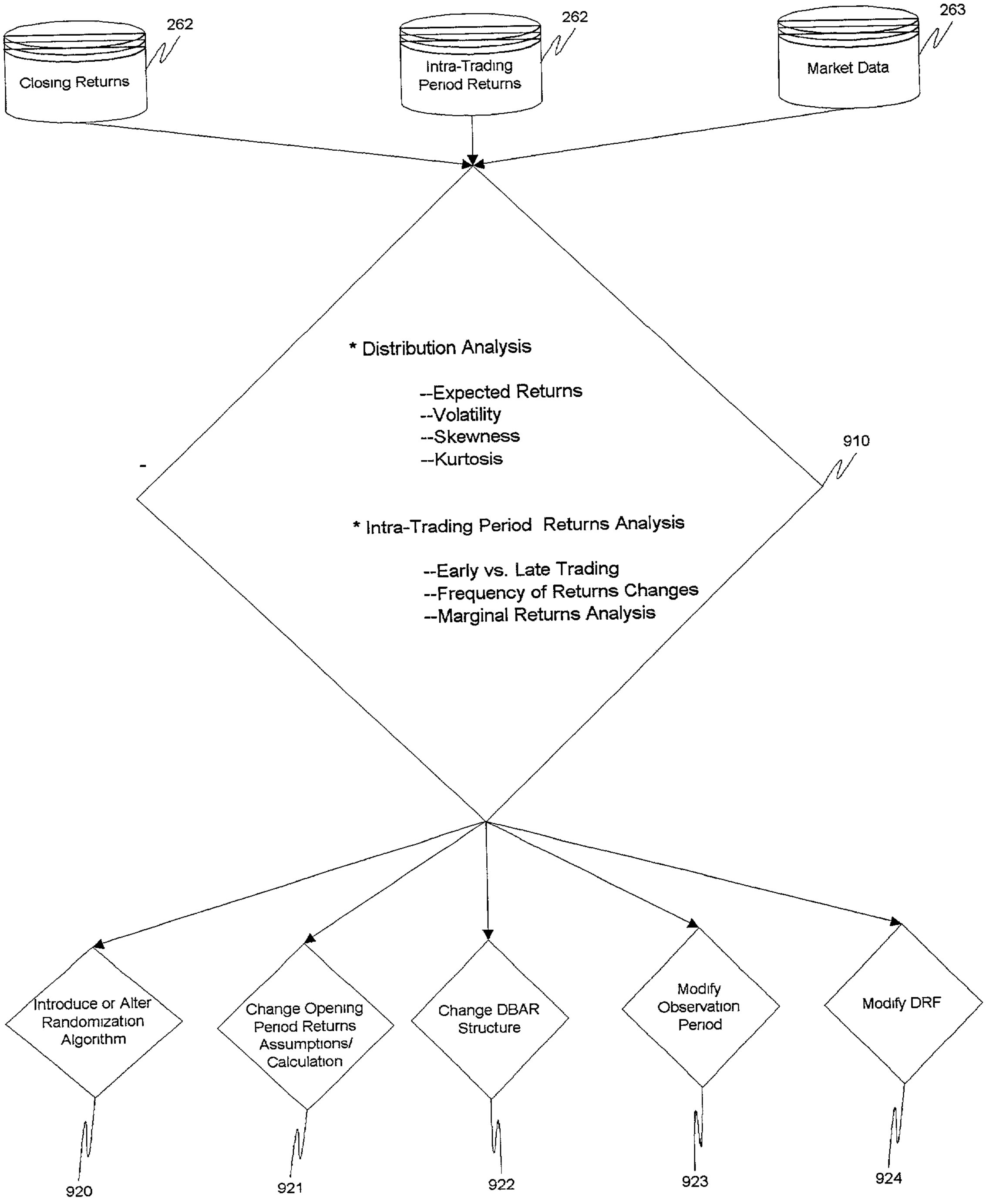
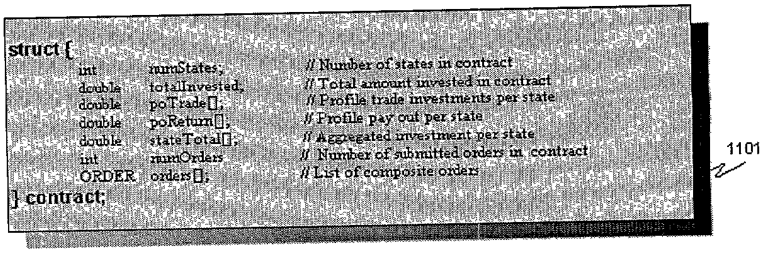
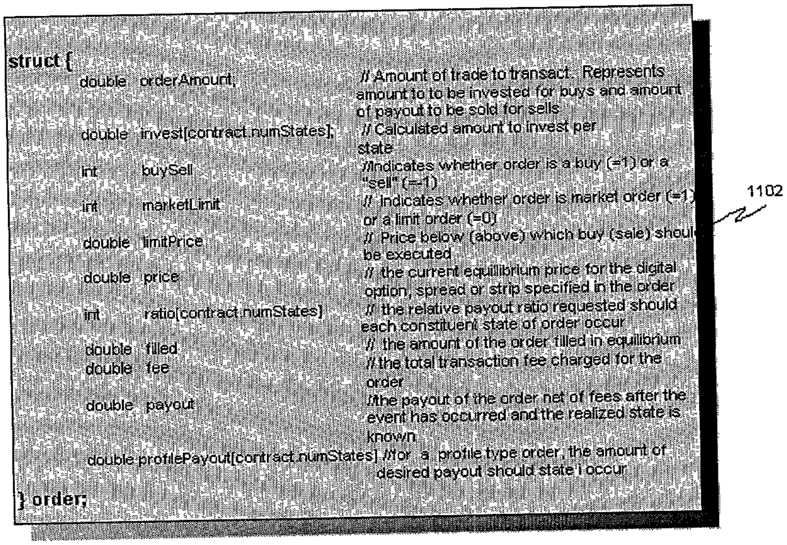

262
Closing Returns
262
Intra-Trading Period Returns
263
Market Data
* Distribution Analysis
--Expected Returns
--Volatility
--Skewness
--Kurtosis
* Intra-Trading Period Returns Analysis
--Early vs. Late Trading
--Frequency of Returns Changes
--Marginal Returns Analysis
910
Introduce or Alter Randomization Algorithm
Change Opening Period Returns Assumptions/ Calculation
Change DBAR Structure
Modify Observation Period
Modify DRF
920
921
922
923
924

FIG. 10

FIG. 11



```
struct {
        int       numStates;          // Number of states in contract
        double    totalInvested,      // Total amount invested in contract
        double    poTrade[];          // Profile trade investments per state
        double    poReturn[];         // Profile pay out per state
        double    stateTotal[];       // Aggregated investment per state
        int       numOrders           // Number of submitted orders in contract
        ORDER     orders[];           // List of composite orders
} contract;
```



```
struct {
        double  orderAmount;                     // Amount of trade to transact. Represents
                                                amount to to be invested for buys and amount
                                                of payout to be sold for sells
        double  invest[contract.numStates];      // Calculated amount to invest per
                                                state
        int     buySell                          //Indicates whether order is a buy (=1) or a
                                                "sell" (=-1)
        int     marketLimit                      // Indicates whether order is market order (=1)
                                                or a limit order (=0)
        double  limitPrice                       // Price below (above) which buy (sale) shoul
                                                be executed
        double  price                            // the current equilibrium price for the digital
                                                option, spread or strip specified in the order
        int     ratio[contract.numStates]        // the relative payout ratio requested should
                                                each constituent state of order occur
        double  filled                           // the amount of the order filled in equilibrium
        double  fee                              // the total transaction fee charged for the
                                                order
        double  payout                           //the payout of the order net of fees after the
                                                event has occurred and the realized state is
                                                known
        double profilePayout[contract.numStates] //for a profile type order, the amount of
                                                  desired payout should state i occur
} order;
```

FIG. 12

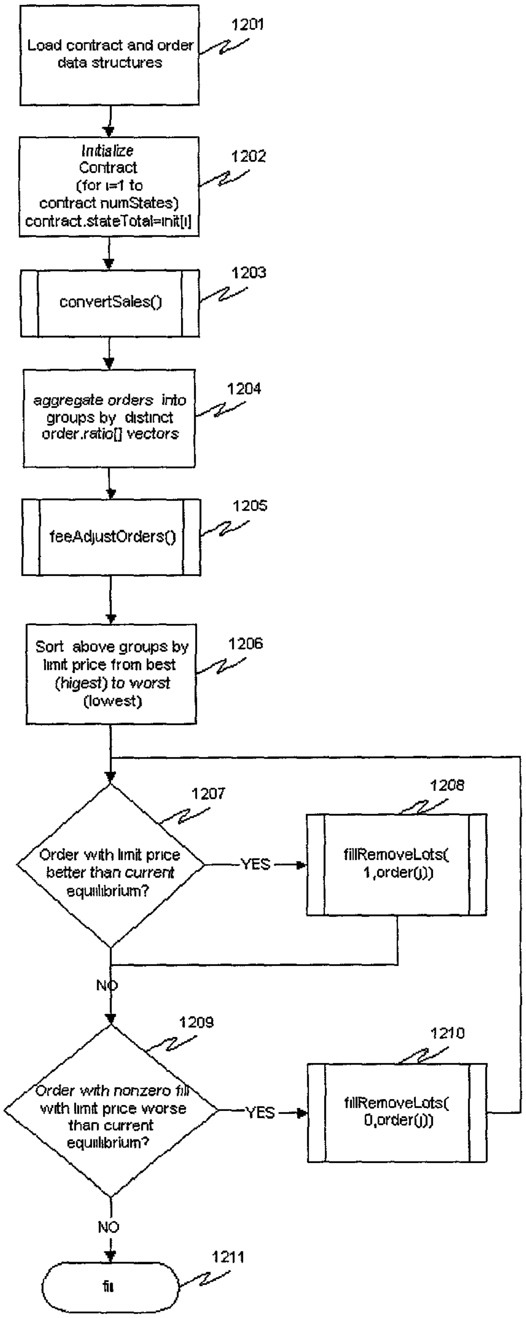

Load contract and order data structures
1201
Initialize
Contract
(for i=1 to
contract numStates)
contract.stateTotal=init[i]
1202
convertSales()
1203
aggregate orders into
groups by distinct
order.ratio[] vectors
1204
feeAdjustOrders()
1205
Sort above groups by
limit price from best
(higest) *to worst*
(lowest)
1206
Order with limit price
better than *current*
equilibrium?
1207
YES
fillRemoveLots(
1,order(j))
1208
NO
Order with nonzero fill
with limit price worse
than current
equilibrium?
1209
YES
fillRemoveLots(
0,order(j))
1210
NO
fin
1211

FIG. 13

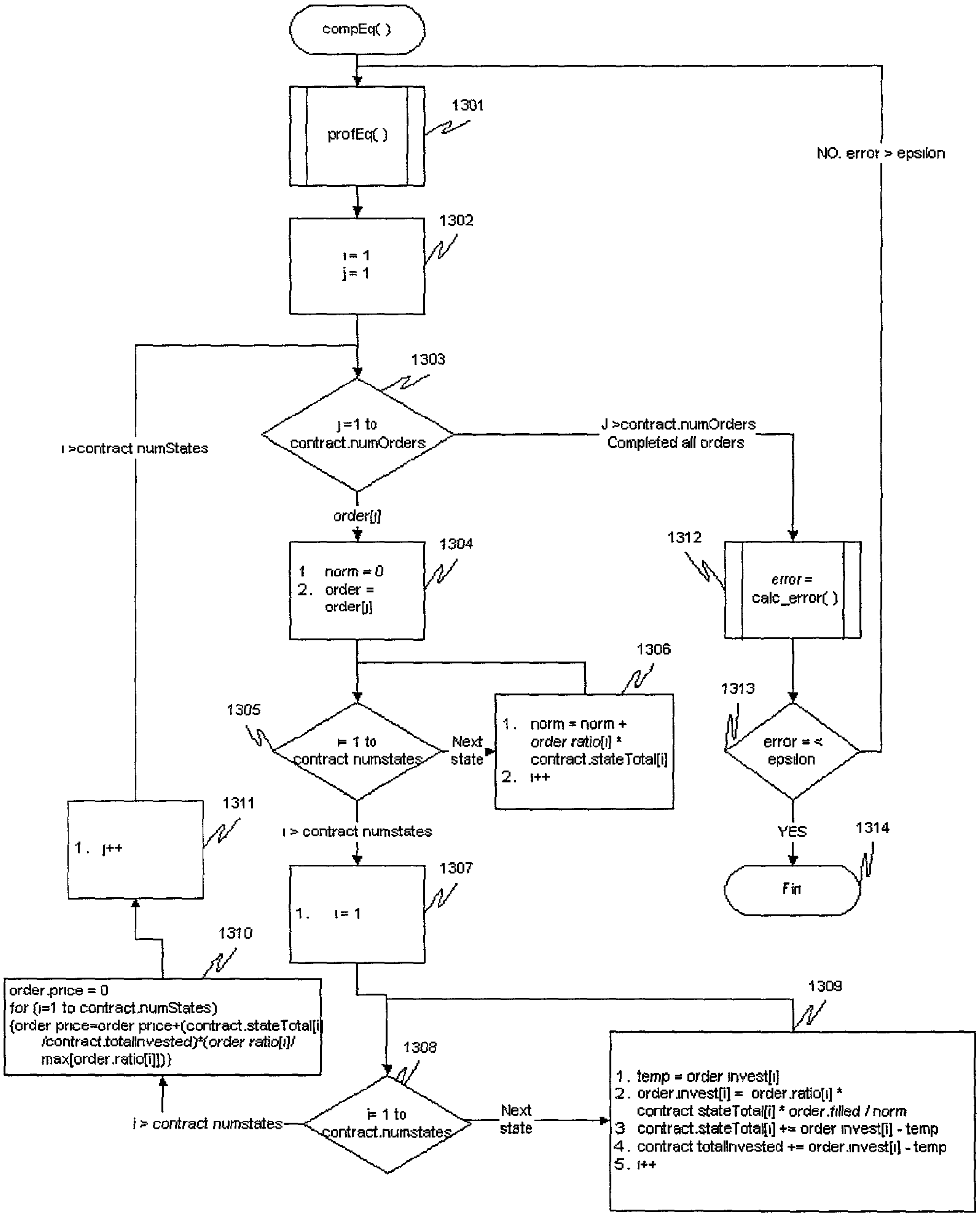

compEq( )
profEq( )
1301
NO. error > epsilon
1302
i = 1
j = 1
1303
j =1 to
contract.numOrders
J >contract.numOrders
Completed all orders
i >contract numStates
order[j]
1304
1 norm = 0
2. order =
order[j]
1312
error =
calc_error( )
1306
1. norm = norm +
order ratio[i] *
contract.stateTotal[i]
2. i++
1305
i= 1 to
contract numstates
Next
state
1313
error = <
epsilon
1311
1. j++
i > contract numstates
1307
1. i = 1
YES
1314
Fin
1310
order.price = 0
for (i=1 to contract.numStates)
{order price=order price+(contract.stateTotal[i]
/contract.totalInvested)*(order ratio[i]/
max[order.ratio[i]])}
1309
1308
i > contract numstates
i= 1 to
contract.numstates
Next
state
1. temp = order invest[i]
2. order.invest[i] = order.ratio[i] *
contract stateTotal[i] * order.filled / norm
3 contract.stateTotal[i] += order invest[i] - temp
4. contract totalInvested += order.invest[i] - temp
5. i++

FIG. 14

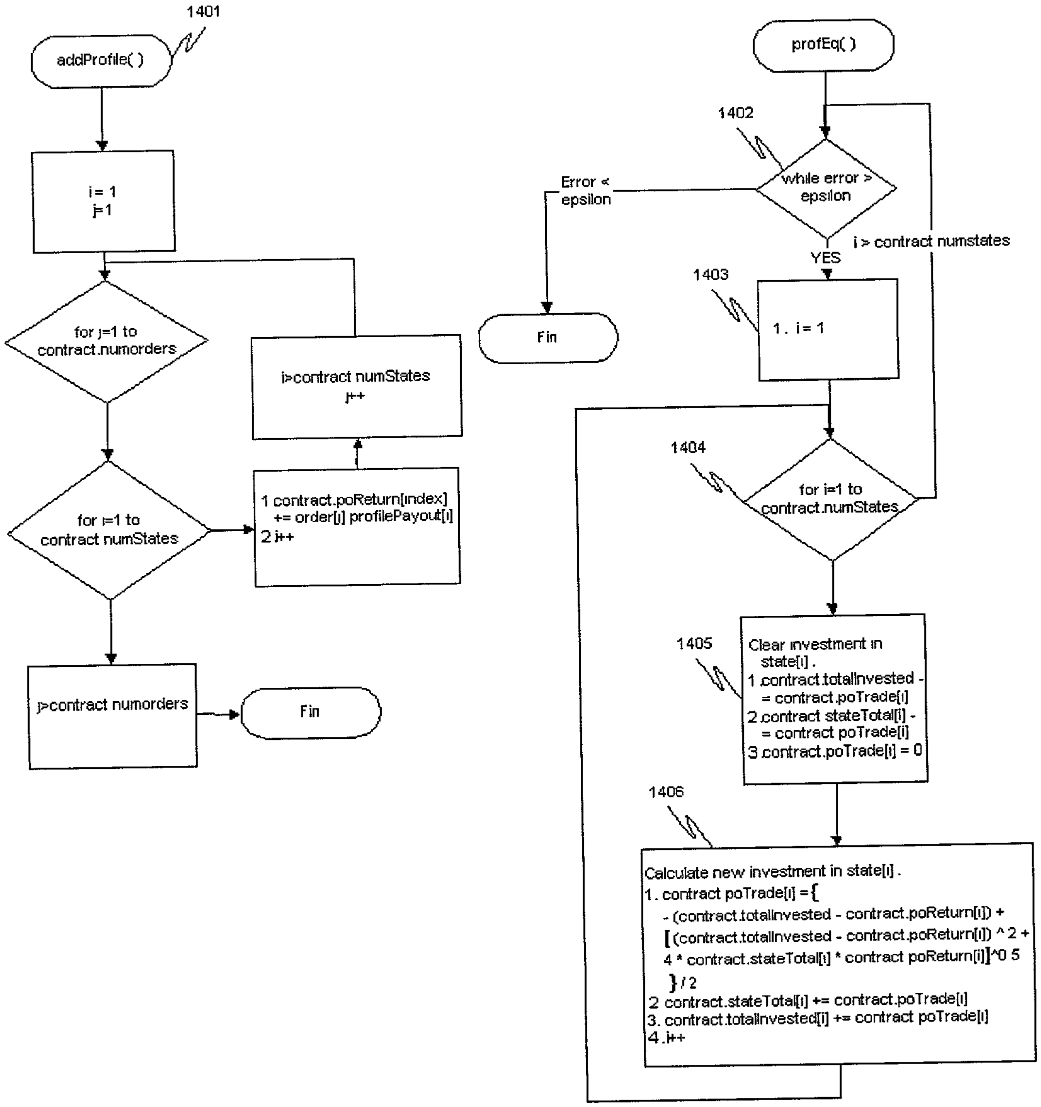

1401
addProfile( )
i= 1
j=1
for j=1 to
contract.numorders
i>contract numStates
j++
for i=1 to
contract numStates
1 contract.poReturn[index]
+= order[j] profilePayout[i]
2 i++
j>contract numorders
Fin
profEq( )
1402
while error >
epsilon
Error <
epsilon
Fin
i > contract numstates
YES
1403
1. i = 1
1404
for i=1 to
contract.numStates
1405
Clear investment in
state[i].
1 contract.totalInvested -
= contract.poTrade[i]
2 contract stateTotal[i] -
= contract poTrade[i]
3 contract.poTrade[i] = 0
1406
Calculate new investment in state[i].
1. contract poTrade[i] = {
- (contract.totalInvested - contract.poReturn[i]) +
[(contract.totalInvested - contract.poReturn[i]) ^ 2 +
4 * contract.stateTotal[i] * contract poReturn[i]]^0 5
}/ 2
2 contract.stateTotal[i] += contract.poTrade[i]
3. contract.totalInvested[i] += contract poTrade[i]
4. i++

FIG. 15

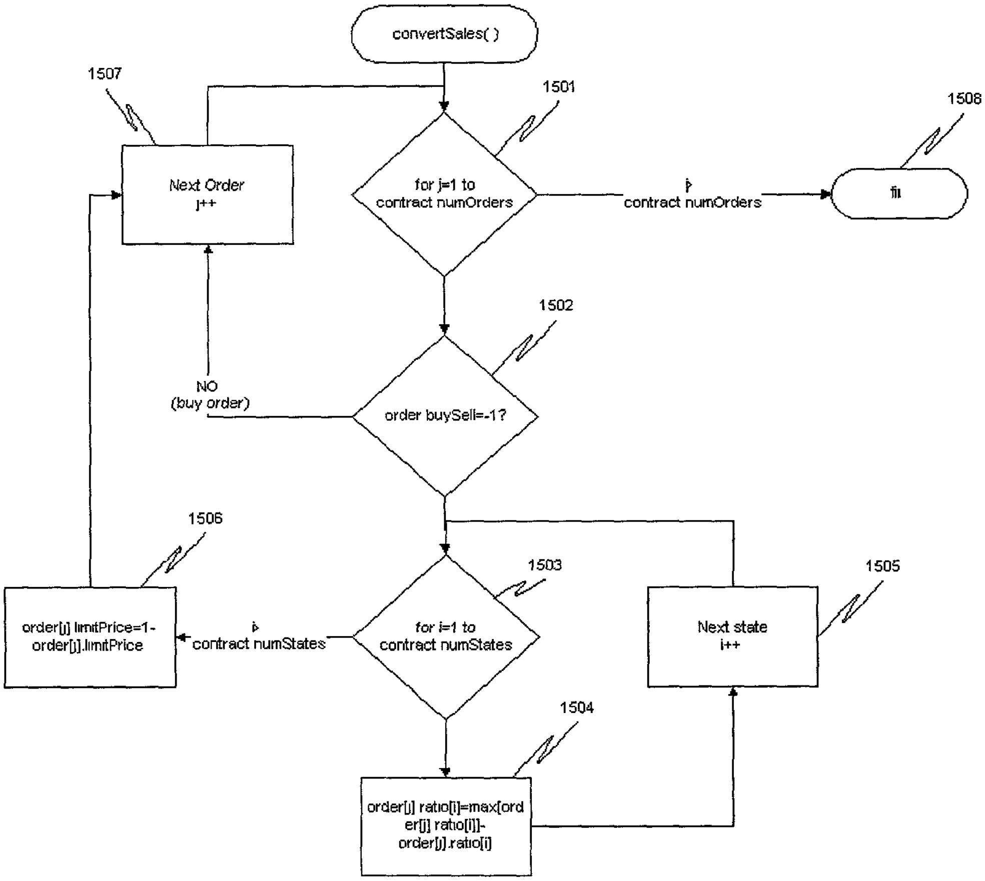

convertSales( )
1507
1501
1508
Next Order
j++
for j=1 to
contract numOrders
>
contract numOrders
fin
1502
NO
(buy order)
order buySell=-1?
1506
1503
1505
order[j].limitPrice=1-
order[j].limitPrice
>
contract numStates
for i=1 to
contract numStates
Next state
i++
1504
order[j].ratio[i]=max[ord
er[j].ratio[i]]-
order[j].ratio[i]

FIG. 16

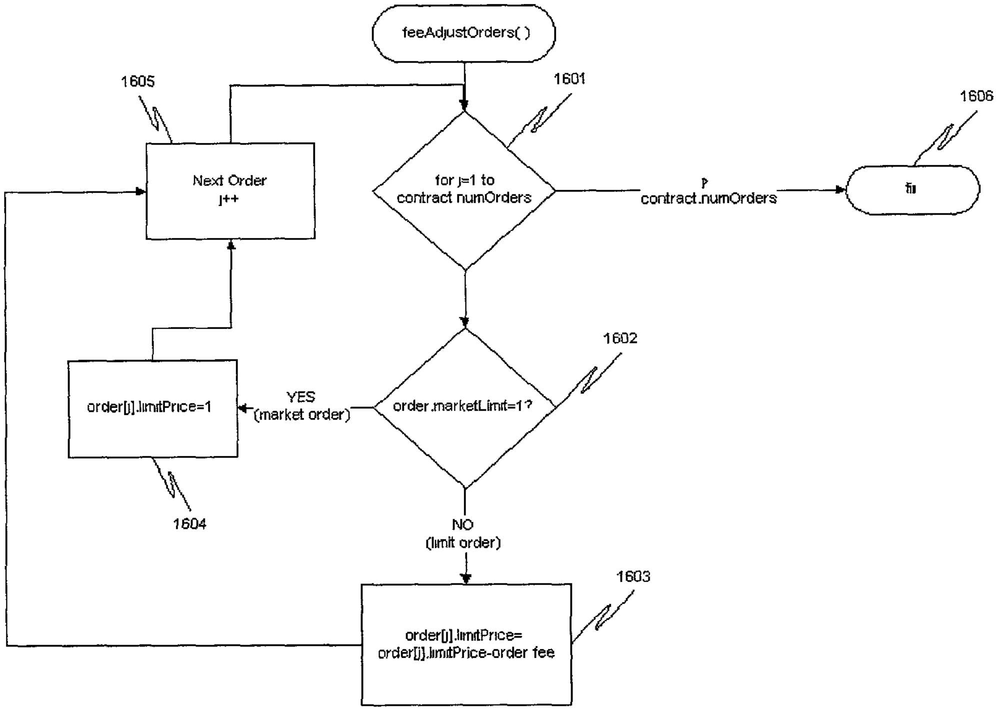

feeAdjustOrders( )
1605
1601
1606
Next Order
j++
for j=1 to
contract numOrders
>
contract.numOrders
fin
1602
order[j].limitPrice=1
YES
(market order)
order.marketLimit=1?
1604
NO
(limit order)
1603
order[j].limitPrice=
order[j].limitPrice-order fee

FIG. 17

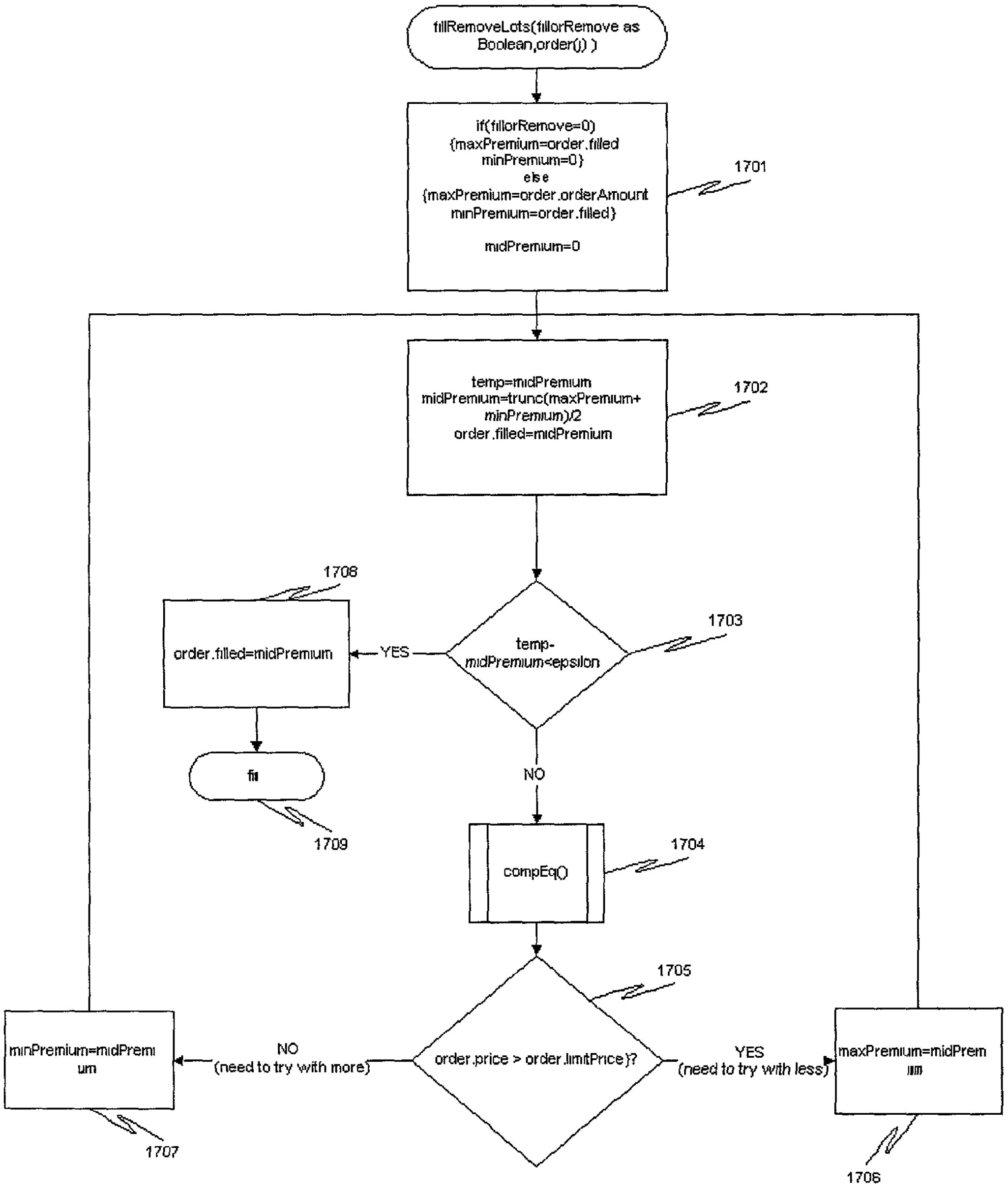
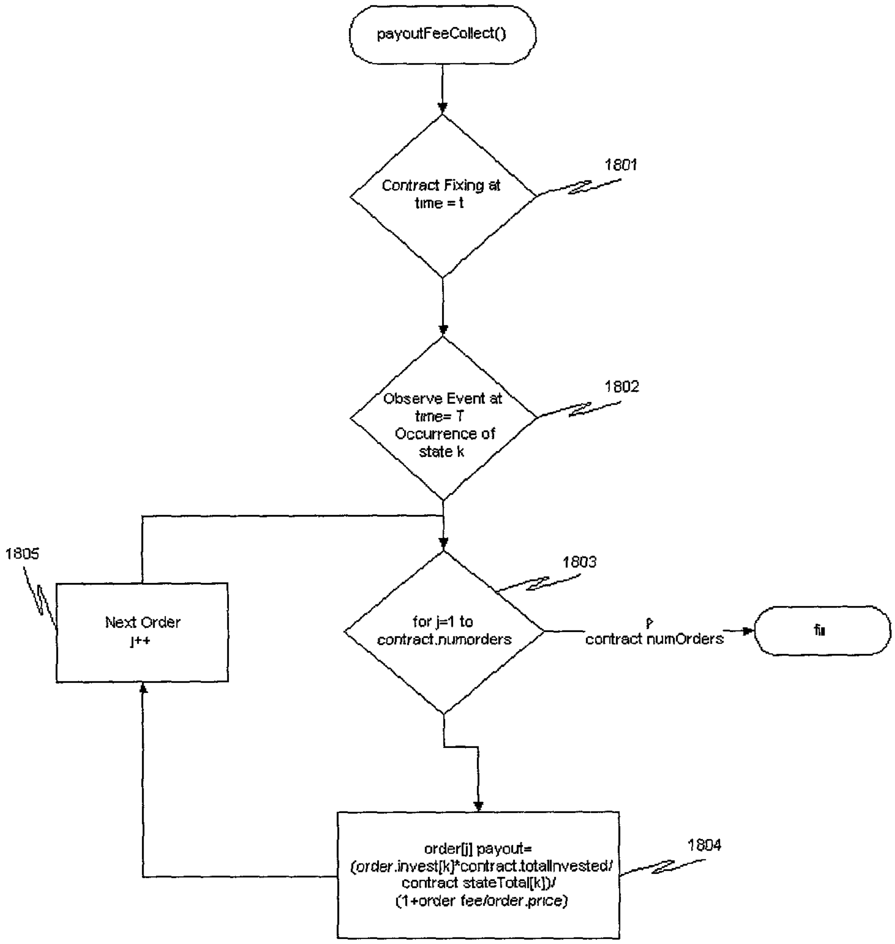

fillRemoveLots(fillorRemove as Boolean,order(j))
if(fillorRemove=0)
{maxPremium=order.filled
minPremium=0}
else
{maxPremium=order.orderAmount
minPremium=order.filled}
midPremium=0
1701
temp=midPremium
midPremium=trunc(maxPremium+
minPremium)/2
order.filled=midPremium
1702
temp-
midPremium<epsilon
1703
YES
order.filled=midPremium
1708
fin
1709
NO
compEq()
1704
order.price > order.limitPrice)?
1705
NO
(need to try with more)
minPremium=midPremi
um
1707
YES
(need to try with less)
maxPremium=midPrem
ium
1706

FIG. 18


payoutFeeCollect()
Contract Fixing at time = t
1801
Observe Event at time= T Occurrence of state k
1802
1805
Next Order j++
1803
for j=1 to contract.numorders
P
contract numOrders
fii
order[j] payout= (order.invest[k]*contract.totalInvested/ contract stateTotal[k])/ (1+order fee/order.price)
1804